내용

[**개발 환경 설정** 2](#_Toc64298316)

[**씬 설명** 2](#_Toc64298317)

[**폴더 설명** 3](#_Toc64298318)

[**프리팹 설명** 3](#_Toc64298319)

[**TrackingImageVisualizer** 5](#_Toc64298320)

[**Canvas** 6](#_Toc64298321)

[**Script** 7](#_Toc64298322)

[BtnDownCheck 7](#_Toc64298323)

[CanvasUI 8](#_Toc64298324)

[CustomTrackingImageController 8](#_Toc64298325)

[CustomTrackingImageVisualizer 8](#_Toc64298326)

[ObjController 8](#_Toc64298327)

# **개발 환경 설정**

* Unity version: Unity 2019.4.20
* Nreal SDK version: 1.5.7

# **씬 설명**

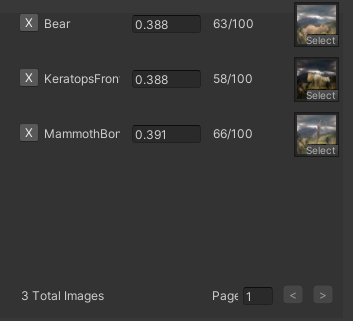
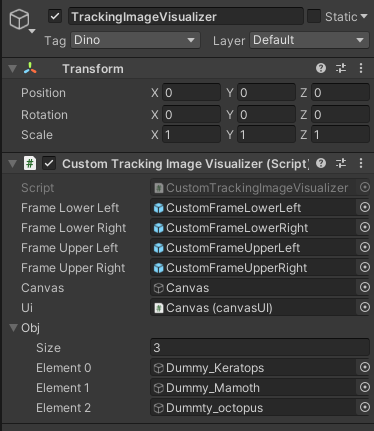
기본적인 구성은 NRSDK->Demo->ImagerTracking 씬을 토대로 만들었습니다.

거기서 Controller 라는 빈 오브젝트를 추가해서 2개의 스크립트로 씬을 구성했습니다.

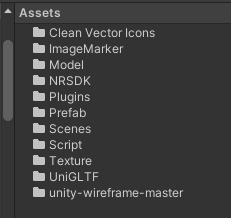
특이사항으로는 UniGLTF라는 에셋을 임포트했습니다.

Glb 모델 파일들을 임포트해주는 에셋인데 고객쪽에서 glb 파일을 임포트를 원하셔서 넣었습니다. 화면 상단에 UniGLTF-1.27이 활성화되어있습니다.

이미지마커 DB의 순서(좌측)대로 인식했을 때 나타나는 오브젝트는 프리팹의 Obj 순서(우측)입니다.

# **폴더 설명**



Clean Vector Icons : UI 만들 때 사용했던 아이콘 모음입니다.

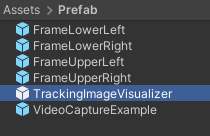
ImagerMarker : 이미지마커를 위한 이미지 모음입니다.

Model & UniGLTF : UniGLTF로 임포트한 glb 모델 파일들을 Model폴더에 저장했습니다.

Unity-wireframe-master : Grid Material 폴더입니다.

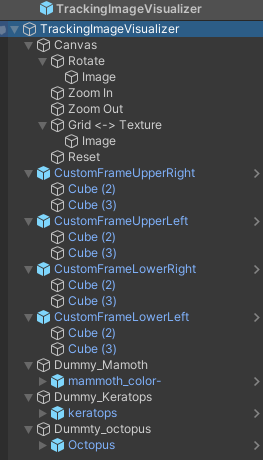
# **프리팹 설명**

프리팹에서 설명드릴 것은 하나 밖에 없습니다.



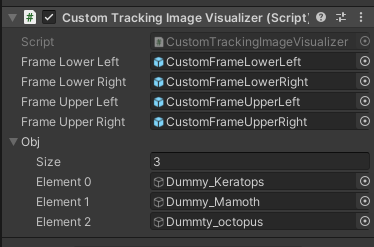
TrackingImageVisualizer 입니다.

프리팹을 보시면 이렇게 되어 있습니다.



하나씩 설명드리겠습니다.

## **TrackingImageVisualizer**



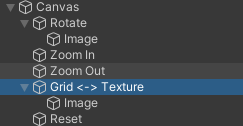
CustomTrackingImageVisulizer 스크립트가 붙어있는데 이것은 기존의 TrackingImageVisualizer를 수정했습니다.

변경한 부분은 Obj 부분입니다.

이미지마커 가장 위에서부터 0, 1, 2 순서로 나가기 때문에 Obj를 리스트로 만들어서 오브젝트를 배치할 수 있게 만들었습니다.

Size를 변경해서 오브젝트를 드래그해서 넣을 수 있게 만들었습니다.

## **Canvas**

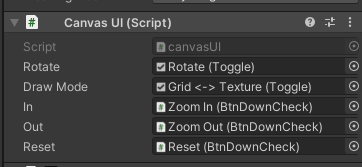
 

캔버스는 총 5개의 버튼으로 구성되어 있습니다.

왼쪽부터 Rotate, Zoom IN, Zoom OUT, GRID <->Textrue, Rest 버튼입니다.

Rotate, Grid <-> Texture 버튼은 Toggle 버튼으로 구성되어 있으며 나머지는 그냥 버튼입니다.

특이사항으로는 Canvas UI 라는 Script를 가지고 있는데 캔버스의 자식들을 부모에서 접근하기 위해 만든 스크립트입니다.

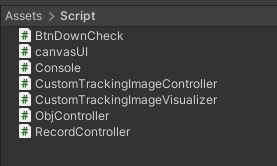


그냥 버튼들은 BtnDownChech 라는 스크립트로 버튼이 눌렸는지 확인합니다.

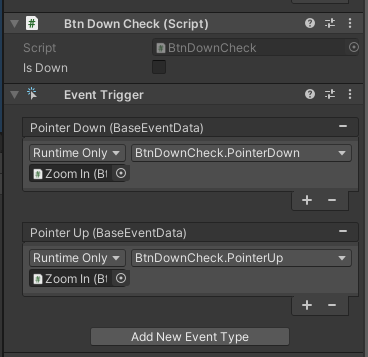
따로 스크립트로 눌렸는지 확인하는 이유는 누르는 동안 줌 인 & 아웃을 구현하기 위해 만들었습니다.

## **Script**

스크립트 폴더 안의 내용물입니다.



### BtnDownCheck



줌 인 & 아웃 버튼에 붙어있는 스크립트입니다. 단순히 버튼이 눌렸는지 체크합니다.

특이사항으로는 Event Trigger 컴포넌트가 필요합니다. 거기서 Pointer Down과 Pointer Up으로 버튼을 누르고 있는지 확인합니다.

### CanvasUI

단순하게 캔버스의 자식인 버튼들을 가지고 있습니다.

줌 인 & 아웃, 리셋 버튼은 BtnDownCheck로 가지고 있는데 이것은 뒤에 설명할 ObjController에서 활용하기 위해 따로 스크립트를 만들었습니다.

### CustomTrackingImageController

기존에 있던 TrackingImageExampleController 스크립트를 조금 수정했습니다.

기존에 있던 스크립트와 기능은 동일합니다.

활성화된 오브젝트를 외부에서 접근하기 위해 프로퍼티를 만들었습니다.

주석대로 벽에 걸려있는 이미지를 인식하더라도 수직으로 서 있게끔 하기 위해서 추가했습니다.

활성화된 오브젝트들을 순차 탐색하면서 전부 비활성화 되어 있다면 UI를 활성화시키는 코드입니다. 예제 씬에서는 한 번 이미지를 인식하면 다른 곳을 쳐다봐도 UI 창이 활성화되지 않아서 추가했습니다.

### CustomTrackingImageVisualizer

기존에 있던 TrackingImageVisualizer 스크립트를 조금 수정했습니다.

인식한 이미지를 바탕으로 오브젝트를 활성화합니다. 위의 컨트롤러에서 인식한 Index를 바탕으로 활성화할 오브젝트를 제외한 나머지를 비활성화 합니다.

### ObjController

활성화된 오브젝트를 컨트롤합니다. 터치를 통해 회전 및 스케일 조정이 가능하고 UI버튼을 눌렀을 때의 처리도 이 스크립트에서 진행합니다. CustomTrackingImageController 에서 만든 프로퍼티에 접근해서 오브젝트를 컨트롤합니다.